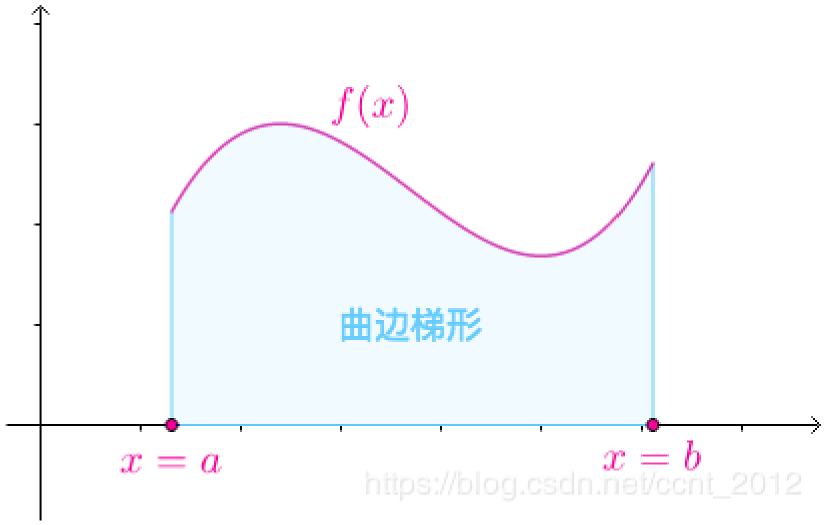
化工应用数学 第三章 数据处理 讲义

（03.数值积分）

定积分：



几何上来说，函数 f(x)的定积分是其与 x 轴之间围出来的曲边梯形的面积（如上图）。

对于一些简单的函数，我们能够通过牛顿-莱布尼茨公式进行计算：

$$\int_a^b f(x)\,dx = F(b) - F(a) = F(x)|_a^b.$$

上式表示一个连续函数在区间[a,b]上的定积分等于它的任意一个原函数在区间[a,b]上的增量。

要注意：*牛顿-莱布尼茨公式仅仅是定积分与不定积分之间数学上的一个计算关系。*

而对于一些较复杂的函数（没有原函数）或者在实验中所得到的列表形式的数据，上述的牛顿-莱布尼兹公式就无法处理了，这时候就需要用到数值的方法对于定积分进行求解。

黎曼和：

数值积分的算法是基于定积分的数学定义来进行的。根据数学定义，函数 f(x)的定积分是其在区间[a,b]上积分和/黎曼和的极限。

积分和/黎曼和：设函数 f(x) 在区间[a,b]上连续，将区间[a,b]分成 n 个子区间 $[x_0,x_1]$, $(x_1,x_2]$, $(x_2,x_3]$, …, $(x_{n-1},x_n]$，其中 $x_0=a$，$x_n=b$。可知各区间的长度依次是：$\Delta x_1=x_1-x_0$，在每个子区间$(x_{i-1},x_i]$中任取一点 ξ_i（1,2,...,n），计算和式：

$$\sum_{i=1}^{n} f(\xi_i)\Delta x_i$$

该和式就是函数的积分和/黎曼和。

黎曼和的限制：

黎曼和在求解数值积分时会遇到下面的限制：

1. 当要求积分的函数是以列表的形式存在的时候，由于没有函数具体形式，我们无法使用黎曼和的方式求解数值积分；
2. 即使函数具有具体的表达式，但对于一些比较畸形的函数来说，使用黎曼和的方式求解数值积分往往需要特别小的步长，从而计算速度上过于缓慢

为了解决上述问题，我们可以采取插值型积分公式进行计算，下面介绍其中的牛顿-柯斯特公式

牛顿-柯斯特公式：

牛顿-柯特斯公式以 Roger Cotes 和艾萨克·牛顿命名，它是以函数于等距 n+1 点的值，使用一个 n 次的多项式来近似原来的函数，再求积分。

求解过程中，我们的目的是将定积分转化为各点函数值的加权求和问题，即：

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

一般我们采用拉格朗日插值多项式作为被积函数的近似，对于积分进行求解，在进行相应数学变换之后，可以得到：

$$\int_I f(t)dt \approx \mathbf{I}_{\mathrm{appr}}(x_1, x_2, \cdots, x_n) = (b-a)\sum_{k=0}^{n} C_k^{(n)} f(x_k)$$

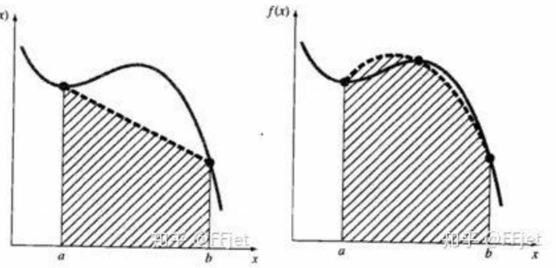
其中：

$$C_k^{(n)} = \frac{1}{n}\int_0^n \prod_{k\neq j}\frac{t-j}{k-j}dt = \frac{(-1)^{n-k}}{n\cdot k!(n-k)!}\int_0^n \prod_{k\neq j}(t-j)dt$$

当 n=1 时，牛顿-柯特斯公式可以表示为：

$$\mathbf{I}_{\mathrm{appr}}(x_1, x_2, \cdots, x_n) = \sum_{i=0}^{n-1}(x_{i+1}-x_i)\left(\frac{f(x_{i+1})+f(x_i)}{2}\right)$$

很容易的，我们可以发现这种计算方式是用梯形的面积取代了我们之前在黎曼和计算中的矩形面积，如下图左所示。



同样的，我们可以发现 n=2 所对应的是使用过三点的二次曲线所围成的曲

边梯形的面积代替了之前的矩形面积（如上图右所示），这时如果我们进一步的取中间节点为左右节点的中点，我们可以得到：

$$\int_a^b P(x)\,dx = \frac{b-a}{6}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

这一公式叫做 Simpson 积分公式。

对于其它 n 取值我们也可以得到相应公式，其对应系数可查阅相关手册中的表格，下面列出了 6 以内 n 值的系数。

n	$C_k^{(n)}$						
1	$\frac{1}{2}$	$\frac{1}{2}$					
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$				
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$			
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$		
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$	
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$

$$\sum_{k=0}^{n} C_k^{(n)} = 1$$

$$C_k^{(n)} = C_{n-k}^{(n)}$$

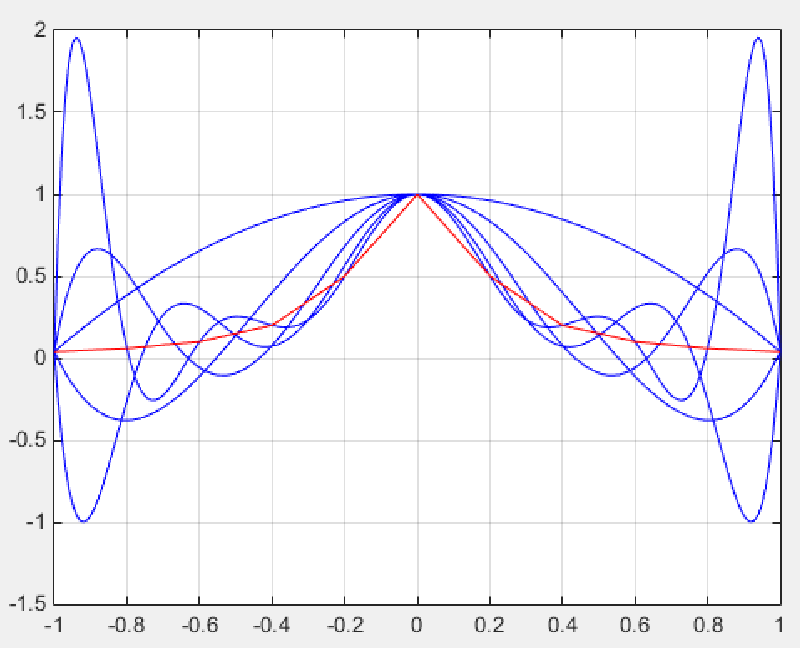
<u>牛顿-柯斯特公式的精度</u>：

求积公式的代数精度：如果某个求积公式对于次数小于 m 的多项式均能准确成立，但对 m+1 次多项式不能准确成立，则称该求积公式具有 m 次的代数精度。

可以证明，n 阶牛顿-科特斯插值型求积公式至少具有 n 次代数精度；而偶数阶牛顿-科特斯插值型求积公式至少具有 n+1 次代数精度。

7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$	
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{28350}$	$\frac{-4540}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

可以看出，越高阶的牛顿-科特斯公式需要的插值点越多，精度也会越高；但是需要注意当n≥8时，牛顿-科特斯系数会出现负数，其计算稳定性得不到保证。这一现象最早是龙格在研究多项式插值的时候发现的，他发现有的情况下，并非取节点越多多项式就越精确。例如 f(x)=1/(1+25x^2)，它的插值函数在高阶插值时两个端点处发生剧烈的波动，造成较大的误差，如下图所示，这一现象也叫龙格现象。



复化求积公式：

因为龙格现象的存在，我们无法使用较高阶的牛顿-科特斯公式，但有时低阶牛顿-科特斯公式无法满足计算的要求，这时候我们可以将积分区间分成若干子区间，然后在每个子区间上使用低阶牛顿-科特斯公式，最后将每个子区间上的积分值相加，以此来计算整个区间上的积分，这种方法称为复化求积法。

复化梯形求积公式：

$$\int_a^b f(x)dx \approx T_n = \frac{h}{2}\left[f(a) + 2\sum_{k=1}^{n-1} f(x_k) + f(b)\right]$$

复华辛普森求积公式：

$$\int_a^b f(x)dx \approx S_n = \frac{h}{6}[f(a) + 4\sum_{k=0}^{n-1} f(x_{k+\frac{1}{2}}) + 2\sum_{k=1}^{n-1} f(x_k) + f(b)]$$

Python 数值积分：

对于有表达式的函数来说，可以使用 Scipy 中的 integrate 模块，导入方式为：

```
from scipy import integrate
```

求解一到三重积分的函数分别为：

```
integrate.quad(func,a,b,args,full_output)
integrate.dblquad(func,a,b,gfun,hfun,args,epsabs,epsrel)
integrate.tplquad(func,a,b,gfun,hfun,qfun,rfun,args,epsabs,epsrel)
```

其中，各参数的含义如下：

func 为运算对象函数，形式为 func(z,y,x)。a、b 对应变量 x 的积分区域，gfun,hfun 对应变量 y 的积分区域，依次类推。gfun、hfun 等的形式应为函数，其

中 gfun、hfun 是自变量为 x 的函数，qfun、rfun 是自变量为 x、y 的函数。args 可选，为传递给 func 的格外参数；full_output 可选，非零则返回积分信息的 dictionary。如果非零，则还会禁止显示警告消息，并将消息附加到输出元组。epsabs 可选，绝对容差直接传递到内部 1-D 正交积分。默认值为 1.49e-8。epsrel 可选，内部 1-D 积分的相对容差。默认值为 1.49e-8。运算结果输出一个元组，第一项为积分结果，第二项为绝对误差，此外还有收敛情况等信息。

```
import numpy as np
from scipy import integrate
def func1(x):
    return x**2
def func2(x,y):
    return x*y
def ybmin(x):
    return 0
def ybmax(x):
    return 2*x+1

result1=integrate.quad(func1,0,1)
print('integrate x**2 from 0 to 1 is:',result1)
result2=integrate.dblquad(func2,0,1,ybmin,ybmax)
print('integrate xy from 0 to 1 is:',result2)
```

结果显示为：

```
integrate x**2 from 0 to 1 is: (0.33333333333333337, 3.700743415417189e-15)
integrate xy from 0 to 1 is: (1.4166666666666665, 4.970732498804344e-14)
```

而对于以列表形式给出的数据，我们可以使用 trapz 和 simps 两个函数进行数值积分，它们分别对应了一阶和二阶牛顿-科特斯公式。

```
import numpy as np
from scipy.integrate import simps
from scipy.integrate import trapz

x=[1,2,3,4,5,6,7,8,9,10]
y=[1,4,9,16,25,36,49,64,81,100]

result=trapz(y,x)
print('integrate with 1st order is:',result)
result=simps(y,x)
print('integrate with 2nd order is:',result)
```

结果显示为：

```
integrate with 1st order is: 334.5
integrate with 2nd order is: 333.16666666666663
```

更多其他数值积分相关函数可以参考 scipy 说明部分：

https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html